The Bus Guardian Design of FlexRay Automotive Communication Systems

Gang-Neng Sung, *Student Member, IEEE*, Ching-Lin Wang, and Chua-Chin Wang†, *Senior Member, IEEE*

Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424
Email: ccwang@ee.nsysu.edu.tw

***Abstract*— This paper presents a Bus Guardian Design used in an in-vehical network compliant with FlexRay standards. FlexRay is a new standard for data/signal communication among electronic devices installed in a vehicle. One of the most important components is the Bus Guardian (BG) in charge of security and safety in the FlexRay standards. The BG proposed in this work is implemented by hardware description language (HDL) and the core area is 880 $\times$ 350 mm^2. The performance is 3.22 mW at a 80 MHz system clock using a typical 0.18 μm CMOS process.**

***Keywords*—FlexRay, baseband, Bus Guardian, automobile electronic, in-vehical networking.**

I. Introduction

Car electronics has been deemed as the 4th "C" right after the Computer, Communication and Consumer electronics. The car electronics include power train, chassis safety, peripheral electronics control system, telematics communication system, in-vehicle networking, etc. Many novel electronic devices have been introduced and installed in recently publicized cars, e.g., car TVs. Therefore, an in-vehicle network has been proposed to control and supervise all of the automobile electronics. Prior in-vehicle networks were mainly composed of CAN (controller area network) or LIN (Local Interconnect Network) which emphasized that safety and reliability. However, the limited 1 Mbps bandwidth is not sufficient for rapid growth of the data/signal in an in-vehicle networking. By contrast, the MOST (Media Oriented Systems Transport) [1] network provided a very efficient mechanism for transporting high volumes of media information, but lack of control capability.

FlexRay V2.1 is the latest communication protocol [2] proposed by several automobile power houses, including BMW, Daimler-Chrysler, General Motos, Freescale, Philips, Robert Bosch, Volkswagen, etc, in spring 2005. It is designed to provide message and data exchange among electronic devices installed in a vehicle. FlexRay will not replace the existing network. By contrast, it can integrate an co-exist with existing network systems, including CAN, LIN, MOST and J1850 protocol, etc. FlexRay requires 10 Mbps data rate in either one of the two channel of an ECU (electronic control unit) [4]. If a single channel is used alone, the speed of the total data rate is expected to be 20 Mbps. Therefore, even the video signals, multimedia and control signals can communicate via the FlexRay network in such a high data rate. The ultimate goal is that the automobile is X-by-wire (X = steer, break, accelerate, A/V, safety, etc.). Table I shows the comparison between FlexRay and prior CAN systems. Fig. 1 shows that the explosive view of a FlexRay network in a vehicle. Fig. 2 shows the block diagram of the ECU composed of Hosts, a Communication Controller (CC), Bus Drivers (BD) and a Bus Guardian (BG).

	CAN	FlexRay
Bit rate	1 Mbps	10 Mbps
Channel	1 channel	2/1 channel (optional)
Network topology	Bus type	Mix. of bus and star type
Communication	Event triggered	Time + Event triggered
Oscillator	Ceramic/Crystal	Crystal oscillator
Network management	Software	Hardware
Network synchronization	Sync segment	Rate compensation

TABLE I
COMPARISON BETWEEN FLEXRAY AND CAN SYSTEM

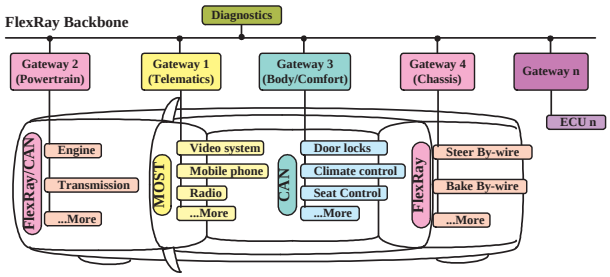

Fig. 1. Explosive view of a FlexRay network

II. Bus Guardian Design

The proposed Bus Guardian is in charge of security and safety for FlexRay communications systems. It protects a channel from interference caused by any message that is not aligned with the communication schedule. In the latest FlexRay Bus Guardian specification, the Bus Guardian and Communication Controller are integrated very compactly. In

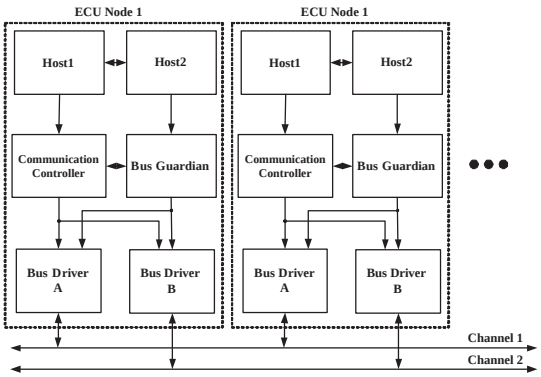

Fig. 2. ECU node on the FlexRay bus

the FlexRay specification, the Bus Guardian is composed of the following function blocks.

- **BG POC:** Bus Guardian POC is the overall control state machine within the BG. The BG POC has a similar functionality as the POC in the CC.
- **Communication controller based functionality:** The BG has an independent clock synchronization mechanism (rate and offset correction and macrotick generation) independent from the local CC. The BG utilizes the same core mechanism as a Communication Controller (Decoder, MAC, FSP (Frame and Symbol Processing) and CSP (Clock Synchronization Process)).
- **CC Supervision (CCS):** This block supervises the slot scheduling of the local CC during wakeup, startup and normal operation and generates the **BGE** signal to enable and disable access for the local CC to the communication medium. If the local CC attempts to access the communication medium when it is not allowed, the BG generates an error message to the Host. The BG disables access to the communication medium depending on the detected error.

The structure of this Bus Guardian specification distinguishes the BG POC operation from the CC supervision process. The BG POC includes all functionalities to integrate the Bus Guardian device into a cluster as an integrating node (configuration, integration ...). The CC supervision (CCS) process performs the supervision of the CC. In order to reduce the power and area consumption, we integrate the Communication-Controller-based functionality block and BG POC block into Communication Controller as shown in Fig. 3.

Fig. 4 shows the signals between all processes and the interface to the CC core mechanism and BD controller. The function of the Bus Guardian is required to supervise the communication controller during different protocol operating modes, including wakeup, communication startup, and synchronized operation. The **TxEN** signal is a transmission enable signal issued by Communication Controller. If the **TxEN** signal is enabled out of the communication schedule, the scheduler denies the bus access and the **BGE** (Bus Guardian enable) signal is not enabled to stop the signal transmitting to the bus.

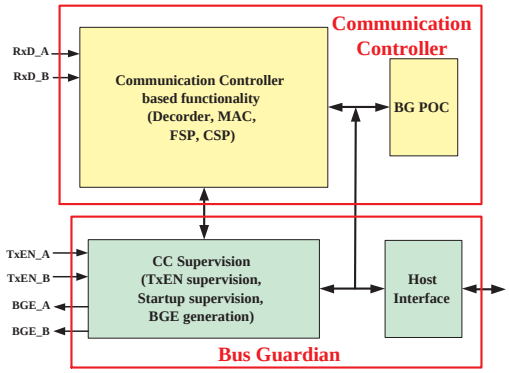

Fig. 3. The block diagram of Bus Guardian

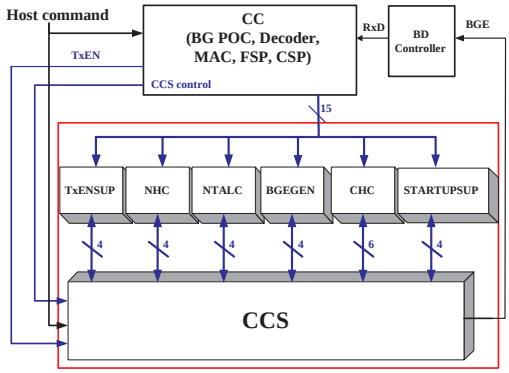

Fig. 4. CCS process signal interface

A. *Communication Controller Supervision (CCS)*

The Communication Controller Supervision process incorporates the core BG functionalities to monitor the correct behavior of the Communication Controller and to control transmit access to the communication medium. The CCS controls the operation of sub-blocks in BG and execute the command form the Host. The Communication Controller Supervision, Host, and Communication Controller can communicate with each other and setup the operation state. As soon as the communication schedule error, supervision error, or operation state error occurs, the CCS will generate an interrupt signal to Host. Fig. 5 indicates an overview of the CCS process.

The state transitions of CCS are described as follows:

1) Host generates a wakeup notification.
2) Wakeup supervision complete, supervision error, frames received or host command notification.
3) Host generates a RUN notification.
4) Supervision error or host command notification.
5) BG POC enters *BG POC:normal active* mode.
6) Transition from *BG POC:normal passive* mode to *BG POC:normal active* mode.

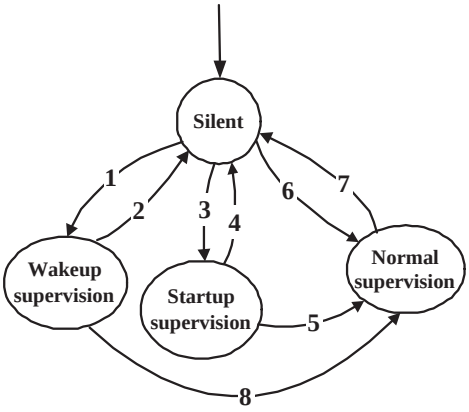

Fig. 5. CCS overview

7) BG POC enters the normal passive mode or BG POC enters ready or halt, supervision error or host CHI CC command notification.
8) BG POC enters *BG POC:normal active* mode.

Every state is explained as follows:

- **Silent:** Silent state is the starting state of the Bus Guardian or initial state when an error occurs. In this state, the communication to the access medium is disabled.
- **Wakeup supervision:** The BG supervises the wakeup behavior of the local CC. Only when no communication on the FlexRay bus is observed, then the BG does allow the CC to transmit wakeup symbol (WUS) on the channel. In this state, **BGE** is enabled and confirmed to transmit a wakeup symbol when **TxEN** is enabled after the pre-setting timing parameter **pdCCSListen Timeout**. If the **TxEN** is not enabled and the wakeup symbol is detected in the channel, BG will generate the interrupt signal to the Host. However, when transmitting the wakeup pattern, BG will stop the transmission after the pre-setting timing parameter **pdCCWakeup Timeout**. Fig. 6 shows the wakeup supervision timing diagram.

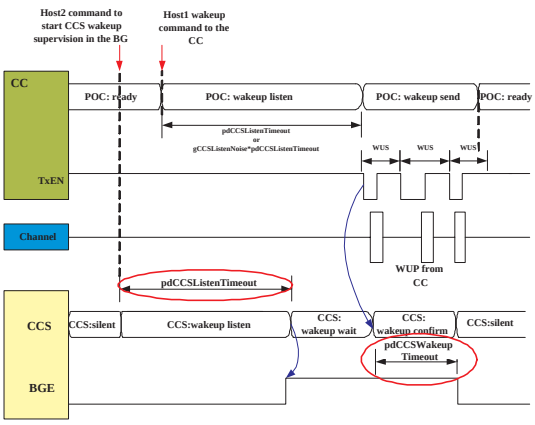

Fig. 6. Wakeup supervision timing diagram

- **Startup supervision:** The BG supervises the startup behavior of the local CC. When the BG detects an erroneous behavior of the local CC, then the silent state is entered. The BG blocks all access to the communication medium for the local CC until the BG POC has been integrated into the cluster and the CCS changes to the normal or supervision state. In this state, BG allows CC transmitting the CAS (collision avoidance symbol) through the channel.
- **Normal supervision:** In the normal supervision state, the BG allows access to the communication medium only for configured assigned slots. When the BG detects an erroneous behavior of the local CC, then the silent state is entered. Fig. 7 shows the **BGE** signal in normal supervision timing diagram.

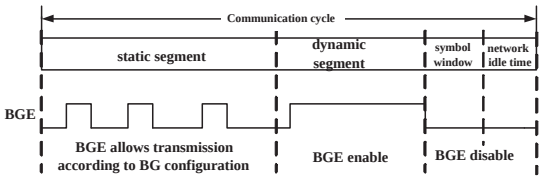

Fig. 7. Startup supervision timing diagram

B. BGEGEN (BGE generation process)

The BGEGEN is in charge of generating the **BGE** signal. The duration of **BGE** signal is controlled by the value of **gdBGEEnable** and **gdBGEDisable**. The **BGE** signal generation process BGEGEN has the following two operating modes:

1) In the GENERATION OFF mode, the **BGE** generation is effectively halted.
2) In the GENERATION ON mode, the **BGE** generation shall be executed. The BG enables transmission to the communication medium for all configured slots according to bus guardian parameters **zsSlotTable.**

Fig. 8 shows the timing diagram of **BGE** signal in BGE generation process.

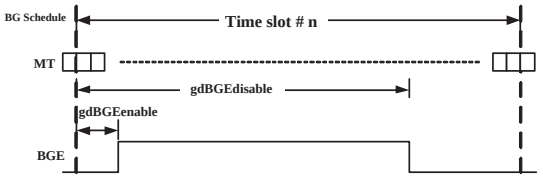

Fig. 8. Startup supervision timing diagram

C. TxENSUP (TxEN supervision process)

The TxEN supervision function checks the correct behavior of the **TxEN** signal with respect to its sequence relationship with the **BGE** signal:

1) The **BGE** signal must be HIGH before any falling edge **TxEN** signal may occur.

2) The **TxEN** signal must be HIGH before any falling edge **BGE** signal may occur.

In addition, the BG checks if there is only one active phase of the **TxEN** signal during one static slot to ensure that only one frame is transmitted by the local CC within one static slot. When the TxENSUP process detects more than one falling edges of the TxEN line within one static slot, then the BG enters *CCS:silent* state. The BG also checks if there is only one falling edge of the TxEN line in one dynamic slot.

D. NHC (Header check in normal supervision)

This process checks if a decoded frame on a channel is received from the local CC or from another node. When the signal Frame decoded on a channel is emitted before the timer **tFrameLength** is expired, then the valid frame is decoded from the local CC and the content of the header is checked.

E. NTALC (TxEN active length check in normal supervision)

The BG checks the active phase length of the TxEN line on the channels. When the detected length of the **TxEN** active phase in not within a defined window, then the *CCS:silent* state is entered.

F. CHC (Header content check process in coldstart)

This process checks if a decoded frame on a channel is received from the local CC or from another node. When the signal Frame decoded on a channel is emitted before the timer **tFrameLength** is expired, then the valid frame is decoded from the local CC and the content of the header is checked. In addition, the CHC process checks if the local CC transmits a CAS only when it is allowed to transmit one.

III. Implementation and Verification

This design is carried out by Verilog HDL code and verified by Xilinx Virtex-4 MB Development Board. The total equivalent gate count of the proposed design is about 4,550. The system clock is 80 MHz and average power consumption is 3.22 mW. Table II gives an comparison with prior work [6]. Fig. 9 shows the overall post-layout simulation by NanoSim. Fig. 10 shows the layout of the Bus Guardian integrate with the Bus Driver. All of the simulations are verified to be compliant with the FlexRay specification [5].

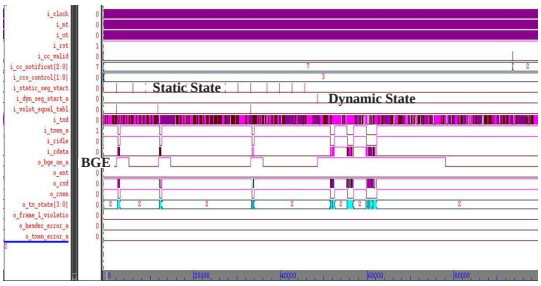

Fig. 9. Post-layout simulation by NanoSim

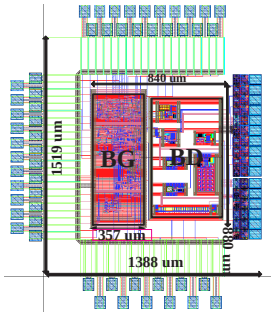

Fig. 10. Layout view of the Bus Guardian and Bus Driver

	Gate Count	Main Clock
[6]	60,000	47 MHz
ours	4,550	80 MHz

TABLE II

Comparison in gate count and clock rate

IV. Conclusion

This paper proposed a hardware implementation of Bus Guardian in the FlexRay automotive communication system using 0.18 μm CMOS cell-based design flow. In the system controller design, we eliminate the Communication-Controller-based functionality block and BG POC block from the original FlexRay specification and integrate them into Communication Controller which can reduce the power and area consumption.

Acknowledgment

The authors would like to express their deepest gratefulness to CIC (Chip Implementation Center) of NAPL (National Applied Research Laboratories), Taiwan, for their thoughtful chip fabrication service. Moreover, this research was partially supported by National Science Council under grant NSC96-2628-E-110-018-MY3. The authors also like to thank "Aim for Top University Plan" project of NSYSU and Ministry of Education (grant no. 96C031001), Taiwan, for partially supporting the research.

References

[1] H. Schopp, and D. Teichner, "Video and Audio applications in vehicles enabled by networked systems," *International Conference on Consumer Electronics,* pp. 218-219, June 1999.

[2] FlexRay Communications System - Protocol Specification V2.1 (http://www.flexray.com), 2005.

[3] A. Techmer, and P. Leteinturier, "Implementing FlexRay on Silicon," *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006 (ICNICONSMCL '06),* pp. 34-40, April 2006.

[4] FlexRay Communications System - Electrical Physical Layer Specification V2.1 (http://www.flexray.com), 2005.

[5] FlexRay Preliminary Node-Local Bus Guardian Specification v2.0.9 (http://www.flexray.com), 2005.

[6] P. M. Szecowka, and M. A. Swiderski, "On hardware implementation of flexray bus guardian module," *International Conference on Mixed Design of Integrated Circuits and Systems 2007 (MIXDES '07),* pp. 309-312, June 2007.